Permutation flowshops with exact time lags to minimize maximum lateness

Julien Fondrevelle, Ammar Oulamara, M C Portmann, Ali Allahverdi

Permutation flowshops with exact time lags to minimize maximum lateness

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2007-IJPR-0659.R1
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	21-Feb-2008
Complete List of Authors:	Fondrevelle, Julien; LIESP Laboratory Oulamara, Ammar; LORIA, ORCHIDS Project; Ecole des Mines de Nancy, I N P L Portmann, M C; Ecole des Mines de Nancy, I N P L Allahverdi, Ali; Kuwait University, Dept of Indl and Mgmt Systems Engineering
Keywords:	FLOW SHOP, SCHEDULING
Keywords (user):	FLOW SHOP, SCHEDULING

International Journal of Production Research
Vol. 00, No. 00, 15 July 2004, 1–21

Permutation flowshops with exact time lags to minimize maximum lateness

J. FONDREVELLE, †,*

A. OULAMARA, M.-C. PORTMANN, ‡

A. ALLAHVERDI §

† LIESP Laboratory, INSA-Lyon, 19 avenue Jean Capelle, F-69621 Villeurbanne, France

‡ ORCHIDS Project LORIA, Ecole des Mines de Nancy, Parc de Saurupt, 54042 Nancy Cedex, France

§ Department of Industrial and Management Systems Engineering, College of Engineering and Petroleum, Kuwait University, P.O. Box 5969, Safat, Kuwait

(*September 2007*)

In this paper, we investigate the m-machine permutation flowshop scheduling problem where exact time lags are defined between consecutive operations of every job. This generic model can be used for the study and analysis of various real situations, which may arise, for instance, in food-producing, pharmaceutical or steel industries. The objective is to minimize the maximum lateness. We study polynomial special cases and provide a dominance relation. We derive lower and upper bounds that are integrated in a branch-and-bound procedure to solve the problem. Three branching schemes are proposed and compared. We perform a computational analysis to evaluate the efficiency of the developed method.

Keywords: Flowshop, exact time lags, maximum lateness, dominance relation, branch-and-bound procedure.

1 Introduction

We consider the problem of scheduling n jobs in an m-machine permutation flowshop where there exist exact time lags between the operations of every job. Each job is processed successively on the machines $1, \ldots, m$ and each machine can process at most one job at a time. Moreover, the time elapsed between every pair of successive operations of the same job must be equal to a prescribed value (exact time lag). We arbitrarily define the time lag between the completion time of the operation on the upstream machine and the starting time of the subsequent operation, processed on the downstream machine (stop-start time lag). Since the processing times are deterministic and known in advance, it is equivalent to consider start-start or stop-stop time lags. When there exists at least one positive exact time lag, permutation schedules, i.e. schedules where the job sequences are the same on all the machines, are no longer dominant, even with two machines (Fondrevelle (2005)). Nevertheless, we consider here only permutation schedules, which are commonly used in industrial applications, for instance, when an automatic conveyor system has to transfer the items from a work station to another one with intermediate buffers managed in a FIFO (First In First Out) rule, job overpassing may be impossible. Moreover, restriction to permutation schedules is sometimes assumed for a simpler production management. The aim is to find a feasible schedule that minimizes the maximum lateness. This objective function is motivated by the fact that it is more general than the makespan and allows to take job due dates into account, corresponding to

*Corresponding author. Email: julien.fondrevelle@insa-lyon.fr

International Journal of Production Research
ISSN 0020-7543 print/ ISSN 1366-588X online
http://www.tandf.co.uk/journals
DOI: 10.1080/00207540xxxxxxxxxxxxx

make-to-order production environments.

The flowshop problem with exact time lags (delays) is a particular case of the flowshop with minimal and maximal time lags. In this situation, the waiting times between the operations are lower- and upper-bounded. Our problem corresponds to the case where for each pair of consecutive operations, the minimal and maximal time lags are equal. In addition, it must be noted that the exact time lag constraints generalize the classical no-wait constraints, for which the waiting time between successive operations equals 0. The no-wait requirement can be found in industries where products must be processed continuously through the stages in order to prevent degradation. Without loss of generality, we consider in this paper the case in which the time lag is an integer value (positive or negative). Let us observe that the case of negative time lags corresponds to job overlapping. This can be used to model a sequence-independent setup time that can be performed while the job is still in process on the preceding machine or a removal time that can be executed while the job is already in process on the succeeding machine (Mitten 1959). Flowshop problems with no-wait and separate setup times occur in several real situations, for instance in chemical, steel or plastic industries (Allahverdi and Aldowaisan 2001). Another example arises when lot-sizing is taken into account. The first item or subset of the lot may be available for processing on a machine before the completion of the last items on the preceding machine. When the exact time lag is positive, the job has to wait for a prescribed amount of time between the machines. This may model a transportation time or an additional processing that does not require any machine. For instance, in manufacturing of thermic paper involving chemical processing, the particular chemical properties required to obtain high quality paper and international standards impose temporal constraints on the process; the same situation arises in pharmaceutical plants, where such constraints must be satisfied at every step, from raw materials and resources preparation to packaging. Similarly Chu and Proth (1996) present an application in an automated laboratory where medical analyses are performed. We could also consider the case of a mechanical company producing high speed gear drives, in which control operations have to be scheduled during a particular time interval: if these control are performed too early, the results may not be significant. On the contrary, late detection of manufacturing defect may lead to high scrap rate. Other practical applications are described in the literature. Hodson *et al.* (1985) study frozen meal production for which maximal delays between cooking and freezing must be respected due to shelf life time and health standards. Kim *et al.* (1996) consider a scheduling problem in a printed circuit board assembly system with lot-sizing; in such process, maximal delays have also to be taken into account to prevent dust deposition on the wafer (Chen and Yang 2006). Several authors focus on the coupled-task problem, which is particularly relevant in the field of pulsed-radar system (Shapiro (1981), Orman and Potts (1997), Ageev and Baburin (2007)). In this problem, the emission and the reception of several electro-magnetic signals have to be processed and the time intervals between these two phases are fixed.

Shop problems with time lags have been extensively studied in the scheduling literature, but in most cases, only minimal time lags are considered (Szwarc (1986), Dell'Amico (1996), Brucker and Knust (1999), Janczewski and Kubale (2001)). Brucker *et al.* (1999) show that various scheduling problems, including flowshop with minimal and maximal time lags, can be reduced to single-machine problems with minimal and maximal time lags between jobs. They propose a branch-and-bound algorithm to minimize the makespan. Finke *et al.* (2002) propose a general model for the two-machine permutation flowshop with minimal time lags and show that this problem can be polynomially solved using an extension of Johnson's algorithm (Johnson 1954). Fondrevelle *et al.* (2006) study the problem of minimizing the makespan in a permutation flowshop with minimal and maximal time lags. Special cases are discussed and a branch-and-bound procedure is developed for the m-machine problem. Concerning the no-wait case, many articles investigate scheduling problems with this constraint. Hall and Sriskandarajah (1996) provide a survey of the research on this topic. From a computational complexity point of

view, the two-machine no-wait flowshop problem of minimizing maximum lateness is shown to be NP-hard (Roeck 1984). This implies that the problem under study is NP-hard as well. The two-machine no-wait flowshop with separate setup times with respect to the maximum lateness is addressed by Dileepan (2004). Only a dominance relation and special cases are provided. Fondrevelle *et al.* (2005) study the same problem where separate removal times are also considered. Special cases are presented and a branch-and-bound algorithm is proposed. To the best of our knowledge, no solution method has been developed for the general problem considered in this paper.

The rest of the paper is organized as follows: Section 2 introduces the notations used and defines different types of jobs. In Section 3, polynomial cases are presented and a dominance relation is proposed for the two-machine problem. In Section 4, lower and upper bounds are developed and integrated in a branch-and-bound procedure with different branching schemes proposed. Finally some computational results are discussed in Section 5.

2 Notations and a preliminary result

In this paper, we use the following notations:

- n: number of jobs
- m: number of machines
- $p_{j,k}$: processing time of job j on machine k
- $\theta_{j,k}$: exact time lag for job j between machine k and machine $k+1$
- $C_{j,k}$: completion time of job j on machine k
- d_j: due date of job j
- $L_j = C_{j,m} - d_j$: lateness of job j

The aim is to determine the job completion times on every machine so that all the constraints are satisfied and the criterion $L_{max} = max\{L_j/1 \leq j \leq n\}$ is minimized. Since the maximum lateness is a regular criterion, semi-active schedules (i.e. left-shifted schedules) are dominant and we will only consider such schedules.

We state the following property, which will be useful in the rest of the paper.

PROPERTY 2.1 *The lateness of each job can be expressed depending on the completion time on machine $k, k < m$, as follows:*

$$L_j = C_{j,k} - d'_{j,k}$$

where $d'_{j,k} = d_j - \sum_{k \leq t \leq m-1}(\theta_{j,t} + p_{j,t+1})$ is the due date for job j on machine k.

Proof The lateness of job j is defined as $L_j = C_{j,m} - d_j$. Due to the exact time lag constraints, the completion time of job j on any machine i can be computed from the completion time on the succeeding machine:

$$C_{j,i} = C_{j,i+1} - p_{j,i+1} - \theta_{j,i}$$

By induction, we have

$$C_{j,k} = C_{j,m} - \sum_{k \leq i \leq m-1} (\theta_{j,i} + p_{j,i+1})$$

which leads to the stated formula (see Figure 1). □

INSERT FIGURE 1 ABOUT HERE

According to the value of each exact time lag, we will distinguish between the following job types:

- The *covering-shape* jobs, for which there exists a machine k such that the processing period on any other machine is included in the processing period on machine k:

$$\forall i,\ 1 \leq i \leq m, C_{j,i} - p_{j,i} \geq C_{j,k} - p_{j,k} \text{ and } C_{j,i} \leq C_{j,k}$$

(see Figure 2). Depending on the machine index k, such a job will be called k-covering-shape.

INSERT FIGURE 2 ABOUT HERE

- The *no-covering-shape* jobs, for which the processing periods on the machines are all disjoint. This corresponds to the case where the exact time lags are non-negative:

$$\forall i,\ 1 \leq i \leq m-1, \theta_{j,i} \geq 0$$

(see Figure 3).

INSERT FIGURE 3 ABOUT HERE

- The *mix-covering-shape* jobs, which do not belong to the previous job classes (see Figure 4).

INSERT FIGURE 4 ABOUT HERE

3 Special case

In this section, we present a polynomial time algorithm for a special case and a dominance relation for the two-machine problem.

3.1 *Polynomial case*

THEOREM 3.1 *If, for a given machine k, all the jobs are k-covering-shape, then an optimal schedule is obtained by using the Earliest Due Date (EDD) rule on the due dates $d'_{j,k}$ on machine k.*

Proof Suppose that all the jobs are k-covering-shape. Consider an arbitrary schedule where the job sequence on machine k is $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$. Due to the definition of k-covering-shape jobs, the earliest starting time for every job will be on machine k and the latest completion time for that job will be on machine k as well. More precisely, the i-th job $\pi(i)$ will be scheduled on machine k between $\sum_{1 \leq h \leq i-1} p_{\pi(h),k}$ and $\sum_{1 \leq h \leq i} p_{\pi(h),k}$. Therefore, the problem is equivalent to a single-machine problem with processing times $p_{j,k}$ and due dates $d'_{j,k}$ on this machine. It is a well known result that EDD provides an optimal schedule for this problem. □

This result generalizes the special cases presented in Fondrevelle *et al.* (2005) for only two machines with separate setup and removal times. These are additional operations that must be performed on the machine respectively before and after the processing of the job. Thus, the machine is busy for the corresponding times and cannot process other operations. During the setup and removal times, the presence of the job on the machine is not required, so that it can be processed on the preceding or following machine. In the case of the no-wait flowshop with such constraints, a job j with processing, setup and removal times on machine k, respectively denoted by $t_{j,k}$, $s_{j,k}$ and $r_{j,k}$, and a due date e_j, can be replaced in our model by a job j with a processing time $p_{j,k} = s_{j,k} + t_{j,k} + r_{j,k}$ on machine k, an exact time lag $\theta_{j,k} = -r_{j,k} - s_{j,k+1}$ between machines

k and $k+1$, and a due date $d_j = e_j + r_{j,m}$.

The special case presented corresponds to a situation when machine k can be considered as the only bottleneck machine. Such situations are known to relate to polynomially solvable cases for several classical problems (see for instance the survey by Monma and Rinnooy Kan (1983) for permutation flowshop problems without time lags to minimize makespan). It could be possible to state other conditions under which the problem studied here can be optimally solved using a simple rule such that EDD. Since these conditions are rather restrictive, we do not present them in this paper.

3.2 *Dominance relations for two-machine problems*

We extend the dominance relations presented by Dileepan (2004) for the two-machine no-wait flowshop with separate setup times to the two-machine flowshop with exact time lags. Consider a sequence $\alpha = (S_1, i, j, S_2)$ where job i precedes immediately job j, and a sequence $\beta = (S_1, j, i, S_2)$ which is identical to α, except that j precedes immediately i (where S_1, S_2 denote partial sequences). The objective is to find conditions under which α dominates β.

As mentioned earlier, the no-wait flowshop with setup times is a particular case of our problem. Following our notations, the conditions proposed in Dileepan (2004) can be expressed as follows:

PROPOSITION 3.2 *Dileepan (2004)*

- *Case A: If*
  - $p_{i,1} + \theta_{i,1} \leq min_{1 \leq u \leq n}\{p_{u,2} + \theta_{u,1}\}$,
  - $p_{j,1} + \theta_{j,1} \leq min_{1 \leq u \leq n}\{p_{u,2} + \theta_{u,1}\}$,
  - $p_{i,2} + \theta_{i,1} \leq p_{j,2} + \theta_{j,1}$
  - *and* $d_i \leq d_j$,

  then solution α *dominates solution* β.

- *Case B: If*
  - $p_{i,1} + \theta_{i,1} \geq max_{1 \leq u \leq n}\{p_{u,2} + \theta_{u,1}\}$,
  - $p_{j,1} + \theta_{j,1} \geq max_{1 \leq u \leq n}\{p_{u,2} + \theta_{u,1}\}$,
  - $p_{j,2} + \theta_{j,1} \leq p_{i,2} + \theta_{i,1}$
  - *and* $d'_{i,1} \leq d'_{j,1}$,

  then solution α *dominates solution* β.

The general ideas used to establish this property are the following:

- Case A: If
  - in solution α there is no idle time on machine 2 during the time interval between the end of S_1 and the completion of j,
  - in solution β there is no idle time on machine 2 during the time interval between the end of S_1 and the completion of i,
  - machine 1 becomes available sooner after j in solution α than after i in solution β,
  - and i has a smaller due date than j on machine 2,

  then solution α dominates solution β.

- Case B: If
  - in solution α there is no idle time on machine 1 during the time interval between the end of S_1 and the completion of j,

- in solution β there is no idle time on machine 1 during the time interval between the end of S_1 and the completion of i,
- machine 2 becomes available sooner after j in solution α than after i in solution β,
- and i has a smaller due date than j on machine 1,

then solution α dominates solution β.

In each case of Property 3.2, the first two conditions are sufficient to avoid idle time as mentioned previously. However, it is possible to state other conditions that are less restrictive and for which the result still holds. These conditions apply to more instances than the previous ones.

Let x denote the last job of the partial schedule S_1 (if S_1 is empty, let $p_{x,1} = p_{x,2} = \theta_{x,1} = 0$). The new conditions can be expressed as follows:

PROPOSITION 3.3

- *Case A': If*
  - $p_{i,1} + \theta_{i,1} \leq min(p_{x,2} + \theta_{x,1}, p_{j,2} + \theta_{j,1})$,
  - $p_{j,1} + \theta_{j,1} \leq min(p_{x,2} + \theta_{x,1}, p_{i,2} + \theta_{i,1})$,
  - $p_{i,2} + \theta_{i,1} \leq p_{j,2} + \theta_{j,1}$
  - *and* $d_i \leq d_j$,

  then solution α dominates solution β.

- *Case B': If*
  - $p_{i,1} + \theta_{i,1} \geq max(p_{x,2} + \theta_{x,1}, p_{j,2} + \theta_{j,1})$,
  - $p_{j,1} + \theta_{j,1} \geq max(p_{x,2} + \theta_{x,1}, p_{i,2} + \theta_{i,1})$,
  - $p_{j,2} + \theta_{j,1} \leq p_{i,2} + \theta_{i,1}$
  - *and* $d'_{i,1} \leq d'_{j,1}$,

  then solution α dominates solution β.

A similar proof to that presented in Dileepan (2004) can be used to demonstrate that in case A' or in case B', solution α dominates solution β.

Let us observe that it is possible to generalize this to a problem with an arbitrary number m of machines, but as m increases, the conditions become more and more complex and restrictive.

4 A branch-and-bound method

In this section, we propose a branch-and-bound algorithm to solve the problem of minimizing the maximum lateness in an m-machine permutation flowshop with exact time lags. As mentioned earlier, we can restrict the search for an optimal solution to semi-active schedules. For a given job sequence π, the optimal placement of the jobs with respect to π on all the machines can be determined polynomially, by scheduling the jobs $\pi(1), \pi(2), \ldots, \pi(n)$ successively, taking into account the time lags constraints. This result is similar to that presented in Fondrevelle *et al.* (2006) and leads us to use a classical scheme based on Ignall and Schrage's method (Ignall and Schrage (1965)). Nodes at depth k of the search tree are associated with initial partial sequences of k jobs. At each separation, a job is added at the end of the current partial sequence. We denote this branching scheme as IS. Another possible scheme for permutation flowshop problems is used in Potts (1980) and generalizes the previous one: an initial and a final partial sequences σ_1 and σ_2 are considered and the branching consists in adding an unscheduled job either at the end of σ_1 or at the beginning of σ_2. We apply two simple versions of the scheme, denoted by PA and PB respectively:

- PA alternatively appends a job to σ_1 and σ_2, depending on the depth of the search tree. Therefore,

for a node at depth $2k$ (respectively $2k+1$), $|\sigma_1| = |\sigma_2| = k$ (respectively $|\sigma_1| = |\sigma_2| + 1 = k+1$), where $|\sigma_i|$ denotes the length of sequence σ_i.

- PB starts by adding the first job (at depth 1) to σ_1, and then sequences successively the other jobs at the beginning of σ_2. Therefore, we always have $|\sigma_1| = 1$ (except at the root node where no job is sequenced).

For both of these schemes, the sequence to which a new job is added at a given depth of the search tree is fixed and depends only on the depth. This property ensures that each complete solution appears exactly once and is associated with one leaf of the search tree.

A depth-first search rule is adopted in the branching procedure. An initial upper bound is provided by the heuristics presented in Section 4.2. The value of the upper bound is then updated each time a new solution with lower objective value is found.

4.1 *Lower bounds*

Lower bounds can be obtained by relaxing some constraints of the problem, so as to reduce it to a simpler, usually polynomially solvable, problem. We identify two special cases of the problem studied here, that can be solved using polynomial algorithm:

- minimizing maximum lateness L_{max} on a single machine, for which the EDD rule is optimal
- minimizing makespan on a two-machine permutation flowshop with exact time lags, for which an optimal sequence is provided by an extension of Gilmore and Gomory's algorithm (Gilmore and Gomory (1964)).

The combination of two-machine permutation flowshop and maximum lateness as objective function leads to NP-hard problems (Lenstra *et al.* (1977), Roeck (1984)), except for very special cases with unit processing times (Bruno *et al.* (1980)). Lower bounds using a relaxation to an NP-hard problem are useful only if there exists an efficient procedure to optimally solve this problem (see for instance Ladhari and Haouari (2005)). We do not consider this possibility here.

Since we study three possible branching schemes, we have to develop lower bounds relatively to these three schemes. We first present the lower bounds associated with IS. Suppose that the initial partial sequence is $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(h))$, in which the first h jobs have been scheduled. The completion times and the lateness of these jobs are exactly determined.

We propose a first lower bound $LB(EDD)$ based on the EDD rule and which makes use of m lower bounds $LB_1, LB_2, \ldots, LB_m$ where $LB_k (k = 1, \ldots, m)$ is computed as follows: for the jobs that have not been scheduled yet, we only take into account the processing on machine k (by relaxing the capacity constraints on all the machines except k and possibly accepting that the operations on these machines might start before time 0). As defined in Section 2, the lateness L_j of each job j can be computed from the completion time on machine k and the due date on this machine, i.e. $L_j = C_{j,k} - d'_{j,k}$. Using a similar argument as in the proof of Theorem 3.1, we could show that the relaxed problem is equivalent to a single-machine problem for the remaining jobs, with processing times $p_{j,k}$ and due dates $d'_{j,k}$ and where the machine becomes available at time $C_{\sigma(h),k}$ (the last job scheduled in the current partial sequence is denoted by $\sigma(h)$). An optimal solution to this problem is provided by EDD applied on $d'_{j,k}$. Let L_k^{EDD} be the corresponding maximum lateness value. Then lower bound LB_k is given by $LB_k = max(L_\sigma, L_k^{EDD})$ where $L_\sigma = max\{L_{\sigma(i)} / 1 \leq i \leq h\}$ denotes the maximum lateness of the current partial schedule. The global lower bound $LB(EDD)$ is defined by $LB(EDD) = max\{LB_k / 1 \leq k \leq m\}$.

We also develop a second lower bound $LB(GG)$ based on a relaxation to a two-machine permutation flowshop problem with exact time lags. For each pair of machines (k_1, k_2), we relax

the capacity constraints of the intermediate machines k ($k_1 < k < k_2$). The machines preceding k_1 or following k_2 are not taken into account. This relaxed problem consists in scheduling the remaining $n - h$ jobs that are not in σ on a permutation flowshop with two machines where each job j has processing times $P_{j,1} = p_{j,k_1}$ and $P_{j,2} = p_{j,k_2}$ and an exact time lag $\Theta_j = \theta_{j,k_1} + \sum_{k_1+1 \leq i \leq k_2-1}(p_{j,i} + \theta_{j,i})$ between the two machines. Moreover each job is assigned a new relaxed due date of constant value $D_j = D = max_{i \notin \sigma}\{d'_{i,k_2}\}$ on the second machine, so that minimizing the maximum lateness is equivalent to minimizing the makespan. The partial schedule associated with σ is taken into account by adding an artificial job $j^\star$ that has to be sequenced first, with processing times $P_{j^\star,1} = C_{\sigma(h),k_1}$ and $P_{j^\star,2} = C_{\sigma(h),k_2} - C_{\sigma(h),k_1}$ and an exact time lag $\Theta_{j^\star} = 0$. We could show that it is possible to impose the sequencing of $j^\star$ in first position by adapting the parameters used to compute the inter-city distances in the traveling salesman problem solved by Gilmore and Gomory's algorithm. Let T be the optimal makespan of the relaxed two-machine permutation flowshop problem (T corresponds to an optimal tour for the associated traveling salesman problem). $T - D$ is a lower bound for the maximum lateness among the unscheduled jobs and $LB_{k_1,k_2} = max(L_\sigma, T - D)$ provides a lower bound for the maximum lateness of all the jobs. Keeping the maximal value for (k_1, k_2) among all $\frac{m(m-1)}{2}$ possibilities leads to the second global lower bound $LB(GG)$. The main disadvantage of this technique comes from the fact that real due dates are not taken into account to get a polynomial relaxed problem.

We propose to adapt the two lower bounds $LB(EDD)$ and $LB(GG)$ to the second and third branching schemes PA and PB. Suppose that the initial partial sequence is $\sigma_1 = (\sigma_1(1), \sigma_1(2), \ldots, \sigma_1(h))$ and the final partial sequence is $\sigma_2 = (\sigma_2(1), \sigma_2(2), \ldots, \sigma_2(h'))$ (where $h' \in \{h-1, h\}$ for PA and $h = 1$ for PB). The completion times and the lateness of the jobs in σ_1 are exactly determined.

For the first lower bound $LB(EDD)$, the procedure for the unscheduled jobs is similar to that for the IS branching scheme: we use successively m relaxations leading to m single-machine problems solved by EDD. Each of this relaxation leads to a lower bound s_k for the starting time of σ_2 on a machine k (which can be used as an earliest starting time): $s_k = C_{\sigma_1(h),k} + \sum_{j \notin \sigma_1 \cup \sigma_2} p_{j,k}$. Thus we can shift the final partial schedule corresponding to σ_2 at the end of this partial schedule so that lower bounds for the lateness of jobs in σ_2 can be computed (due to the exact time lag constraints, the final partial schedule σ_2 can be constructed independently and then shifted at the end of the previous subschedule).

The second lower bound $LB(GG)$ also proceeds similarly as in the IS branching scheme: the first objective is to obtain a lower bound s_k for the starting time of σ_2 on each machine k and then to shift the corresponding partial schedule at the end. Such earliest starting times can be computed by minimizing makespan on a two-machine permutation flowshop with exact time lags, considering an artificial job $j^\star$ associated with σ_1 and the $n - h - h'$ unscheduled jobs, and applying the same relaxations as previously:

$$s_1 = C_{\sigma_1(h),1} + \sum_{j \notin \sigma_1 \cup \sigma_2} p_{j,1} \text{ and for } k > 1, s_k = max_{1 \leq q < k}\{C^{GG}_{max}(q,k)\}$$

where $C^{GG}_{max}(q,k)$ denotes the makespan provided by Gilmore and Gomory's algorithm on the instance corresponding to the unscheduled jobs plus the initial artificial job $j^\star$ associated with σ_1, on machines q and k. The final partial schedule corresponding to σ_2 is then shifted according to the earliest starting times s_k. The use of $LB(GG)$ combined with PB branching scheme (and in a smaller extent with PA) can be motivated by the fact that it leads to better estimations of the earliest starting times of σ_2 compared to the ones obtained with $LB(EDD)$.

4.2 *Upper bounds*

For $1 \leq k \leq m$ we define the heuristic H_k as follows: apply EDD on the due dates $d'_{j,k}$ on machine k and construct the corresponding schedule. The best criterion value obtained can be used as an initial upper bound and will be denoted by H_{EDD}.

We also propose to improve this value through an iterative procedure based on NEH method (Nawaz *et al.* (1983)). The principle of this frequently used scheme is as follows: starting from an initial job list, the schedule is constructed step by step by successively inserting the jobs of the list at the best position in the partial sequence, so as to minimize the objective function. We choose to apply NEH iteratively a prescribed number of times: at each iteration, the final sequence obtained at the previous iteration is used as initial job list. The sequence with the best value throughout the iterations is kept as solution. Depending on the criterion used to construct the initial job list, we define three heuristics $NEH(TT)$, $NEH(JL)$ and $NEH(H_{EDD})$:

- In $NEH(TT)$, the initial list is sorted in decreasing order of total processing time $\sum_{1\leq k\leq m} p_{j,k}$ of jobs.
- In $NEH(JL)$, the initial list is sorted in decreasing order of total length $\sum_{1\leq k\leq m-1}(p_{j,k}+\theta_{j,k})+p_{j,m}$ of jobs.
- In $NEH(H_{EDD})$, the initial job list corresponds to the best sequence provided by the heuristic H_{EDD}.

5 Computational results

We conducted computational experiments to evaluate the performance of the proposed solution procedure. In the case of two machines, we also compared this new general approach with the one used in Fondrevelle *et al.* (2005). Our algorithms were coded in C, and the computational experiments were run on a PC Pentium, 1.2 GHz.

We first used the same instances as in Fondrevelle *et al.* (2005) as benchmarks (classes 1 to 9). These 10-instance classes correspond to two-machine no-wait flowshop problems with separate setup and removal times, which were shown to be particular cases of our problem (mix-covering-shape jobs only, with partial covering between every couple of successive operations).

We also generated new benchmark classes for 5-machine problems, according to the classification given in Section 2. Each class contains 10 instances and the number of jobs is set up to 16, except for class 12 in which $n = 14$. The size of the instances is fixed to avoid excessive time-consuming computational experiments.

- Class 10 corresponds to no-covering-shape jobs only, the processing times of which are randomly drawn between 20 and 50. The time lags are in the interval $[0, 100]$.
- Class 11 corresponds to mix-covering-shape jobs only, the processing times of which are randomly drawn between 20 and 50. $\theta_{j,k}$ is generated between $-p_{j,k+1}$ and 0 so as to have partial covering between every couple of successive operations of the jobs.
- Classes 12 and 13 correspond to covering-shape jobs only. For each job j, a random integer is drawn between 1 and 5 to determine the machine k_j such that j is k_j-covering-shape. The processing times on all the machines except k_j are generated between 20 and 50, and the time lags that are not related to k_j are in the interval $[-20, 20]$. p_{j,k_j}, θ_{j,k_j-1} and θ_{j,k_j} are computed such that j is k_j-covering-shape. Although such problems do not correspond to real situations, it could be interesting to apply our solution method to them in order to evaluate its efficiency in these cases. Class 12 differs from class 13 on only one point: the number of jobs, which is 14 in class 12 while it is set up to 16 in class 13.

Following the method proposed in Potts and Van Wassenhove (1982), we generated the due dates in a range $[P \times x, P \times y]$, where P is a lower bound on the makespan and $x = 1 - T - R/2$, $y = 1 - T + R/2$. T is the tardiness factor, which was set to 0.6 and R is the due date range set to 0.75.

We first compared the performance of the heuristics presented in Section 4.2 on the whole set of instances. In order to set the number of iterations for all the NEH-based heuristics at a satisfactory level, we ran them with a limit of 100 iterations and kept the iteration for which the best value was reached on every instance. The results are shown in Table 1 in which the frequencies of instances for different levels are indicated. Iteration 0 corresponds to the solution given by the initial sequence (without applying NEH insertions scheme). We can note that the original version of the procedure, i.e. with only one iteration, provides the best solution for about 1 instance out of 4. Since the best value is obtained within the first 5 iterations in more than 2/3 of the cases, we chose to execute the NEH-based heuristics with a limit of 5 iterations in the computational experiments. For each instance, the relative error (in %) between the solution found by the heuristic considered and the optimal solution, obtained with the branch-and-bound procedure without time limit, was computed. The average values for each class are given in Table 2. The column *Best* indicates the values provided by the best heuristic for each instance. The lowest average values among the heuristics (excluding the column *Best*) are presented in bold underlined, and the other "good" values (at 1% range) are in bold only. Since the CPU times for the heuristics are very small (less than 0.1 second), we do not report them here.

INSERT TABLE 1 ABOUT HERE

INSERT TABLE 2 ABOUT HERE

As can be seen from Table 2, H_{EDD} is outperformed by the iterative NEH-based methods. This result holds for every instance. $NEH(H_{EDD})$ is slightly outperformed by $NEH(TT)$ and $NEH(JL)$, the average relative errors of which are in the same range and do not increase with the number of machines. Moreover, for each of these heuristics, there exists at least one instance in each class on which the heuristic dominates the two others. As the execution times of these heuristics are very small, we chose to perform the three NEH-based methods and kept for each instance the best value obtained. The corresponding average relative errors are reported in the column *Best*.

As mentioned in the previous section, we tested 6 different versions of our branch-and-bound procedure, depending on the branching scheme and the lower bound used. These 6 configurations are presented in Table 3.

INSERT TABLE 3 ABOUT HERE

To evaluate the quality of the proposed methods, we performed them on each instance with a computational time limit of 1200 seconds. For each class and each version i, we report in Table 4 the number N_i of problems for which the algorithm achieved the time limit and the average computational time t_i (in seconds) for the problems optimally solved before the time limit.

INSERT TABLE 4 ABOUT HERE

Table 4 clearly indicates that the lower bound $LB(GG)$ based on a two-machine reduction is extremely outperformed by $LB(EDD)$, which uses a reduction to a single-machine problem. For most of the instances (120 out of 130) version 2 of the branch-and-bound procedure is not able to find the optimal solution within the time limit, whereas version 1 optimally solves almost all of them (126 out of 130). If we except class 9 (respectively classes 4 and 9), the average computational time for version 4 (respectively version 6) is much larger than the one of version 3 (respectively version 5). A preliminary analysis on the number of nodes in the search tree for which the lower bound is calculated was carried out for versions 1 and 2 but is not reported here. It shows that the

huge difference in performance between the two lower bounds is mainly due to the complexity of $LB(GG)$ compared to $LB(EDD)$, which is much simpler. The average time per node of $LB(GG)$ is approximately 20 times larger than that of $LB(EDD)$ for two-machine problems, which is due to the numerous operations required in the Gilmore and Gomory's algorithm, whereas the EDD rule only needs one sorting. This gap between the two lower bounds can be amplified or attenuated depending on the instance: for some of them, $LB(GG)$ requires much more nodes than $LB(EDD)$ to reach the optimal solution, whereas in other cases the reverse holds. Concerning 5-machine problems, the computations performed in $LB(GG)$ could have been restricted to fewer couples of machines to reduce the computational effort, but as we can notice for classes 1 to 9, this effort is already too huge for 2-machine problems, when there is only one couple of machines.

In order to compare the proposed branching schemes, we have to make a distinction between the classes according to their number of machines. As far as the 2-machine problems are concerned, PB outperforms the other schemes. This could be due to the criterion considered in this study. Indeed IS scheme, which considers the partial sequence of the first jobs, was originally designed for makespan minimization, whereas for maximum lateness criterion, the sequence of the last jobs processed has a greater impact on the solution quality: jobs with a small due date, if sequenced at the end of the schedule, lead to a poor quality. Since such situations are explored earlier with PB, this scheme is more efficient than the IS and PA. Yet, for 5-machine instances, IS appears to be more appropriate. When the number of machines increases, shifting σ_2 at the end of the schedule becomes more complex, which leads to greater computational times for PA and PB. Therefore, we suggest version 5 (PB+$LB(EDD)$) of our branch-and-bound procedure should be used to solve 2-machine problems, whereas version 1 (IS+$LB(EDD)$) should be preferred for its simplicity with a larger number of machines.

Among the first 9 instance classes, 3 seem to be harder to solve than the others: classes 3, 4 and 9. This may be due to the size of the setup times, which are in a range twice as large as the processing times. As far as the 5-machine problems are concerned, it seems that problems with covering-shape jobs only are more difficult to solve than problems with no-covering-shape jobs only or problems with mix-covering-shape jobs only.

The impact of the dominance relation on the resolution is illustrated in Table 5, where we report the average computational time t_{dom} (in seconds) when the dominance relation is taken into account (only for two-machine problems). We also give the corresponding values obtained with the method proposed in Fondrevelle *et al.* (2005), with a prime symbol. Since the branching scheme applied in this method is similar to IS, we consider version 1 of our branch-and-bound algorithm for a fair comparison. We also add the results obtained with version 5 as it is the most efficient version for these problem classes. It is important to note that the dominance test in Fondrevelle *et al.* (2005) is more restrictive than the one we use here and concerns only classes 8 and 9, for which there are no removal times.

INSERT TABLE 5 ABOUT HERE

If we compare the performance of our new branch-and-bound procedure and that of the one proposed by Fondrevelle *et al.* (2005), we can note a significant improvement in computational time: all the two-machine problems (except one for version 1) are optimally solved by the new algorithm and the average computational time is divided by a factor between 5 and 250 except for class 5. The corresponding percentage reduction (between 81.6% and 99.6%) appear in the table. It could be surprising that a method developped for a more general problem outperforms a solution approach dedicated to a particular case. Such a gain is partly due to the improvement of the lower bound. Moreover, the dominance relation, which is more frequently used than the previous one, appears to perform quite well since it results in saving more than 30% of the computational time in average.

We also conducted another series of experiments to evaluate the influence of the number of machines m on the performance of the branch-and-bound procedure (version 1). Since the number of jobs n is directly related to the number of possible solutions to the problem ($n!$), its impact was not studied. Three new classes denoted by 11A, 11B and 11C were generated similarly as class 11, with m equal to 2, 10 and 15 respectively. Table 6 presents the number N_1 of problems (out of 10) for which the algorithm achieved the time limit (1200 seconds) and the average computational time τ (in seconds) when no time limit is imposed. Additionnally, we report the average number Q of nodes evaluated (i.e. how many times the lower bound is computed), $\rho = \tau/Q$ and $\lambda = \rho/m$. Therefore ρ corresponds to the average time to evaluate one node (in seconds). Without time limitation, the maximum CPU time was 3720 seconds for an instance with 15 machines. Although these figures may not be meaningful when comparing them with other methods run on different machines, they can be used as indicators for comparison between instances.

INSERT TABLE 6 ABOUT HERE

We can consider the value of λ as constant since it varies from 0.43×10^{-6} to 0.55×10^{-6}. This does not only hold on average, but also for every instance. By definition, this means that the average time for the branch-and-bound algorithm to evaluate one node increases linearly with the number of machines, which is in agreement with the computational complexity of the lower bound ($O(m)$). Besides, the number of nodes Q seems also to be roughly linear in m. This empirical result needs to be confirmed or contradicted by further experiments. The increase in the number of visited nodes is partially explained by the fact that the lower bound becomes less tight as the number of machines grows.

6 Conclusion

We study the problem of minimizing maximum lateness in m-machine permutation flowshops with exact time lags. These time constraints generalize the classical no-wait constraint and may be used to model no-wait problems with separate setup and removal times. A branch-and-bound method is proposed to solve optimally this NP-hard problem. Three different branching schemes and two lower bounds are tested. The computational results show that for the two-machine problem an efficient method is obtained by progressively building a partial sequence from the end of the schedule, with a lower bound based on the EDD rule. When the number of machines increases, the classical Ignall and Schrage's scheme appears to be more appropriate. The resulting procedure outperforms a previous algorithm and may be improved significantly by using a dominance relation in case of two-machine problems. A natural extension of this work would consist in introducing additional constraints such as release dates, which can easily be integrated in our procedure. Moreover the study of other lateness or tardiness criteria, such as total weighted tardiness, seems to be a promising direction for further research.

7 Acknowledgements

The authors would like to thank the anonymous referees for their valuable comments that help improve the quality of the paper.

REFERENCES

Ageev, A.A. and Baburin, A.E., Approximation algorithms for UET scheduling problems with exact delays. *Operations Research Letters* 2007, **35**, 533–540.

Allahverdi, A. and Aldowaisan, T., Minimizing total completion time in a no-wait flowshop with sequence-dependent additive changeover times. *Journal of the Operational Research Society* 2001, **52**, 449–462.

Brucker, P., Hilbig, T. and Hurink, J., A branch and bound algorithm for a single-machine scheduling problem with positive and negative time-lags. *Discrete Applied Mathematics* 1999, **94**, 77–99.

Brucker, P. and Knust, S., Complexity results for single-machine problems with positive finish-start time-lags. *Computing* 1999, **63**, 299–316.

Bruno, J., Jones III, J.W. and So, K., Deterministic scheduling with pipelined processors. *IEEE Transactions on Computers* 1980, **29**, 308–316.

Chen, J.-S. and Yang, J.-S., Model formulations for the machine scheduling problem with limited waiting time constraints. *Journal of Information and Optimization Sciences* 2006, **27**, 225–240.

Chu, C. and Proth, J.-M., Single machine scheduling with chain structured precedence constraints and separation time windows. *IEEE Transactions on robotics and automation* 1996, **12**, 835–844.

Dell'Amico, M., Shop problems with two machines and time lags. *Operations Research* 1996, **44**, 777–787.

Dileepan, P., A note on minimizing maximum lateness in a two-machine no-wait flowshop. *Computers and Operations Research* 2004, **31**, 2111–2115.

Finke, G., Espinouse, M.-L. and Jiang, H., General flowshop models: job dependent capacities, job overlapping and deterioration. *International Transactions in Operational Research* 2002, **9**, 399–414.

Fondrevelle, J., Résolution exacte de problemes d'ordonnancement de type flowshop de permutation en présence de contraintes d'écarts temporels entre opérations (*in French*). PhD thesis, Institut National Polytechnique de Lorraine, France, 2005.

Fondrevelle, J., Allahverdi A. and Oulamara, A., Two-machine no-wait flowshop scheduling problem to minimize maximum lateness with separate setup and removal times. *International Journal of Agile Manufacturing* 2005, **8**(2), 165–174.

Fondrevelle, J., Oulamara, A. and Portmann, M.-C., Permutation flowshop scheduling problems with maximal and minimal time lags. *Computers and Operations Research* 2006, **33**, 1540–1556.

Gilmore, P.C. and Gomory, R.E., Sequencing a one state-variable machine: a solvable case of the traveling salesman problem. *Operations Research* 1964, **12**, 655–679.

Hall, N. and Sriskandarajah, C., A survey of machine scheduling problems with blocking and no-wait in process. *Operations Research* 1996, **44**, 510–525.

Hodson, A., Muhlemann, A.P. and Price, D.H.R., A microcomputer based solution to a practical scheduling problem. *Journal of the Operational Research Society* 1985, **36**, 903–914.

Ignall, E.J. and Schrage, L.E., Application of the branch and bound technique to some flow-shop scheduling problems. *Operations Research* 1965, **13**, 400–412.

Janczewski, R. and Kubale, M., Scheduling unit execution time tasks with symmetric time-lags. *Journal of Applied Computer Science* 2001, **9**, 45–51.

Johnson, S.M., Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly* 1954, **1**, 61–68.

Kim, Y.-D., Lim, H.-G. and Park, M.-W., Search heuristics for a flowshop scheduling problem in a printed circuit board assembly process. *European Journal of Operational Research* 1996, **91**, 124–143.

Ladhari, T. and Haouari, M., A computational study of the permutation flow shop problem based on a tight lower bound. *Computers and Operations Research* 2005, **32**, 1831–1847.

Lenstra, J.K., Rinnooy Kan, A.H.G. and Brucker, P., Complexity of machine scheduling problems. *Annals of Discrete Mathematics* 1977, **1**, 343–362.

Mitten, L.G., Sequencing n jobs on two machines with arbitrary time lags. *Management Science* 1959, **5**, 293–298.

Monma, C.L. and Rinnooy Kan, A.H.G., A concise survey of efficiently solvable special cases of the permutation flow-shop problem. *R.A.I.R.O. Recherche Oprationnelle* 1983, **17**, 105–119.

Nawaz, M., Enscore Jr, E.E. and Ham, I., A heuristic algorithm for the m-machine, n-job flow shop sequencing problem. *Omega* 1983, **11**, 91–95.

Orman, A.J. and Potts, C.N., On the complexity of coupled-task scheduling. *Discrete Applied Mathematics* 1997, **72**, 141–154.

Potts, C.N., An adaptive branching rule for the permutation flow-shop problem. *European Journal of Operational Research* 1980, **5**, 19–25.

Potts, C.N. and Van Wassenhove, L.N., A decomposition algorithm for the single machine total tardiness problem. *Operations Research Letters* 1982, **1**, 177–181.

Roeck, H., Some new results in flowshop scheduling. *Zeitschrift fur Operations Research* 1984, **28**, 1–16.

Shapiro, R.D., Scheduling coupled-tasks. *Naval Research Logistics Quarterly* 1981, **28**, 489–497.

Szwarc, W., The flow shop problem with time lags and separated setup times. *Zeitschrift fur Operations Research* 1986, **30**, B15–B22.

Table 1. Number of iterations of NEH-based heuristics leading to the best solution

Iterations	$NEH(H_{EDD})$	$NEH(TT)$	$NEH(JL)$	Average
0	3.9%	0%	0%	1.3%
1	17.7%	32.3%	25.4%	25.1%
0-5	71.5%	74.6%	67.7%	71.3%
6-10	22.3%	16.9%	18.5%	19.2%
11-100	6.2%	8.5%	13.9%	9.5%

Table 2. Average percentage relative errors of the heuristics

Class	H_{EDD}	$NEH(H_{EDD})$	$NEH(TT)$	$NEH(JL)$	*Best*
1	22.1	11.9	**8.6**	**8.9**	7.2
2	24.3	9.1	**6.3**	7.3	5.8
3	17.7	4.1	**2.4**	**2.9**	1.8
4	12.9	**4.9**	**5.1**	**4.2**	3.7
5	9.8	4.8	**3.3**	4.8	3.1
6	32.9	12.5	**10.9**	**10.9**	9.0
7	25.9	10.3	12.9	**8.6**	6.9
8	29.1	13.7	**8.0**	**8.4**	6.9
9	14.8	**4.1**	**3.3**	5.3	2.9
10	38.9	**5.6**	**5.8**	**5.3**	4.2
11	33.5	11.5	**7.4**	9.4	7.2
12	26.2	7.1	**5.5**	**6.0**	4.3
13	32.5	9.1	**6.4**	8.0	5.9
Average	24.7	8.4	**6.6**	**6.9**	5.3

Table 3. Configurations for the versions of the branch-and-bound procedure tested

Version	Branching scheme	Lower bound
1	IS	$LB(EDD)$
2	IS	$LB(GG)$
3	PA	$LB(EDD)$
4	PA	$LB(GG)$
5	PB	$LB(EDD)$
6	PB	$LB(GG)$

Table 4. Performance of the branch-and-bound procedures

Scheme	IS				PA				PB			
Bound	$LB(EDD)$		$LB(GG)$		$LB(EDD)$		$LB(GG)$		$LB(EDD)$		$LB(GG)$	
Class	N_1	t_1	N_2	t_2	N_3	t_3	N_4	t_4	N_5	t_5	N_6	t_6
1	0	13.8	9	729.7	0	6.0	2	209.6	0	2.6	0	69.3
2	0	13.2	7	391.1	0	6.8	0	141.6	0	2.1	0	52.8
3	0	22.4	9	499.3	0	26.7	2	74.4	0	16.1	1	16.9
4	0	13.6	8	310.0	0	20.7	0	164.3	0	13.5	0	0.8
5	0	35.5	10	/	0	3.4	2	201.9	0	0.9	0	20.1
6	0	0.4	10	/	0	1.2	0	115.5	0	0.4	0	8.3
7	0	3.7	10	/	0	6.0	4	243.4	0	0.8	0	95.2
8	0	8.9	8	830.5	0	4.1	1	92.6	0	2.0	0	13.4
9	1	10.2	9	202.3	0	70.8	0	25.7	0	35.4	0	0.11
10	0	155.3	10	/	4	255.2	8	648.5	4	229.5	5	660.0
11	0	115.7	10	/	0	335.6	8	818.5	0	187.2	5	418.4
12	0	71.2	10	/	0	307.8	10	/	0	284.3	9	653.3
13	3	209.4	10	/	8	708.5	10	/	7	749.3	9	411.0

Table 5. Performance of the dominance relation and comparison with the method of Fondrevelle *et al.* (2005)

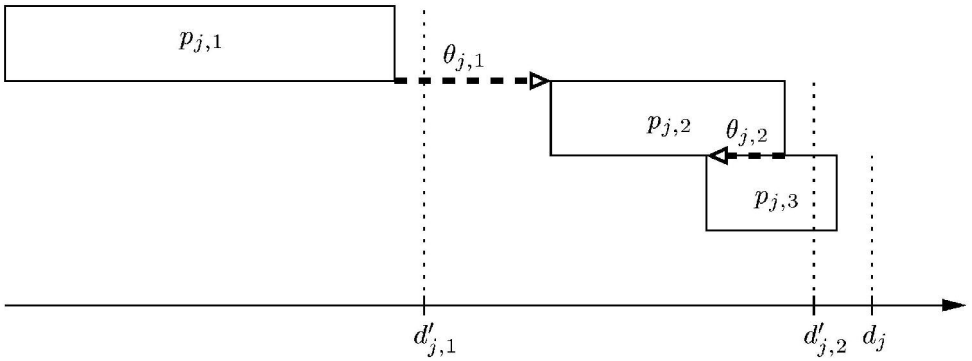
Class	N_1	t_1	t_{1dom}	N'	t'	t'_{dom}	$\frac{t'-t_1}{t'}$ (%)	N_5	t_5	t_{5dom}
1	0	13.8	9.2	1	107.1	/	87.1	0	2.6	1.6
2	0	13.2	10.7	1	171.6	/	92.3	0	2.1	1.6
3	0	22.4	19.4	2	121.6	/	81.6	0	16.1	11.9
4	0	13.6	8.5	4	251.7	/	94.6	0	13.5	7.5
5	0	35.5	14.2	1	59.7	/	40.5	0	0.9	0.6
6	0	0.4	0.3	0	99.7	/	99.6	0	0.4	0.3
7	0	3.7	2.9	1	88.2	/	95.8	0	0.8	0.5
8	0	8.9	6.5	0	166.4	153.6	94.7	0	2.0	1.5
9	1	10.2	5.7	3	182.5	139.7	94.4	0	35.4	20.3

Table 6. Influence of m on the computational time

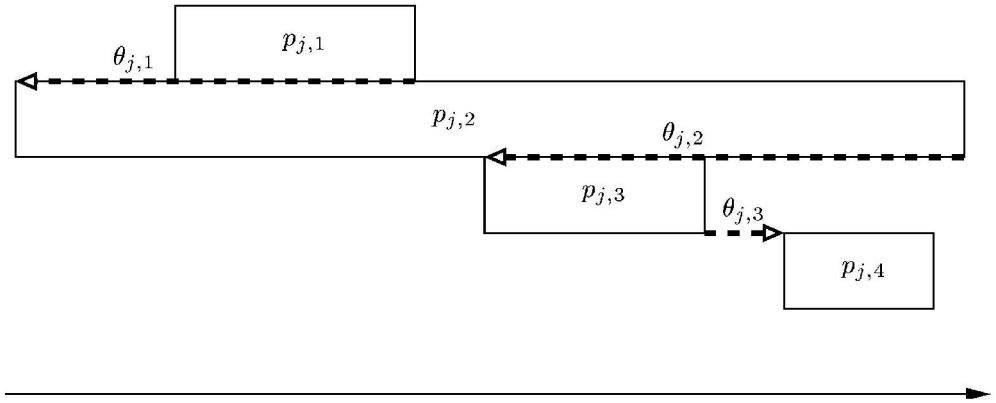
Class	m	N_1	τ	Q	$\rho = \tau/Q$	$\lambda = \rho/m$
11A	2	0	5.6	5.086×10^6	1.10×10^{-6}	0.55×10^{-6}
11	5	0	115.7	47.50×10^6	2.44×10^{-6}	0.48×10^{-6}
11B	10	1	500.9	111.85×10^6	4.48×10^{-6}	0.45×10^{-6}
11C	15	5	1449.9	223.35×10^6	6.49×10^{-6}	0.43×10^{-6}

List of figure captions

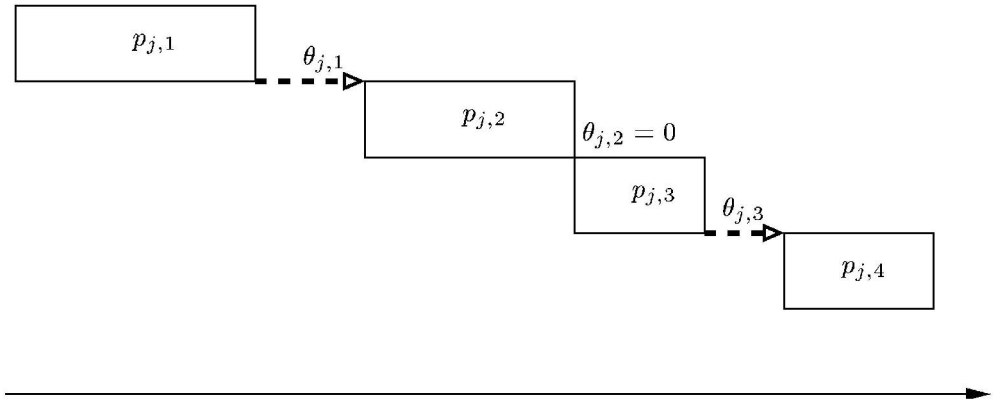
- Figure 1. Due date on each machine
- Figure 2. Covering-shape job
- Figure 3. No-covering-shape job
- Figure 4. Mix-covering-shape job



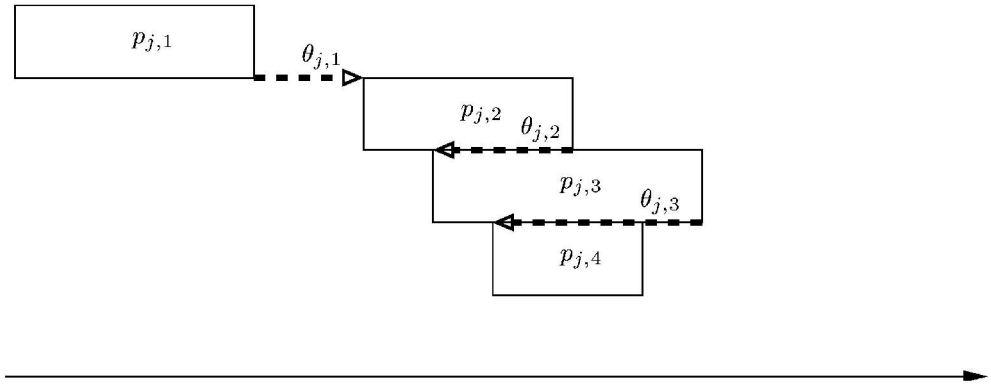
Due date on each machine
105x38mm (600 x 600 DPI)



Covering-shape job
107x43mm (600 x 600 DPI)



No-covering-shape job
107x43mm (600 x 600 DPI)



Mix-covering-shape job
112x43mm (600 x 600 DPI)